Prepare your work folder

```
# Update the Materials folder to the latest version of our GitHub
cd your-path/Programming/Materials
git pull

# Create a Week-13 folder in Assignments
cd your-path/Programming/Assignments
mkdir Week-13
cd Week-13

# Copy the contents of Materials/Week-6 into Assignments/Week-13
cp -R ../../Materials/Week-13/* .
```

Tidyverse

Programming Psychology Experiments (CORE-1)

Barbu Revencu & Maxime Cauté

Session 13 | 10 December 2025

The plan for today

1. `tidyverse`: data manipulation basics
2. `ggplot`: data plotting basics

Data manipulation basics

The structure of the data analysis file

1. **Data manipulation**: The part where you clean your dataset and add all the relevant information for data analysis
2. **Data visualization**: Plotting the results of your experiment
3. **Data analysis**: Inferential statistical tests of your hypotheses

Why data manipulation

Suppose you have collected your experimental data

Depending on many factors (experimental settings, data collection platform), the output data might not be in the right format for analysis

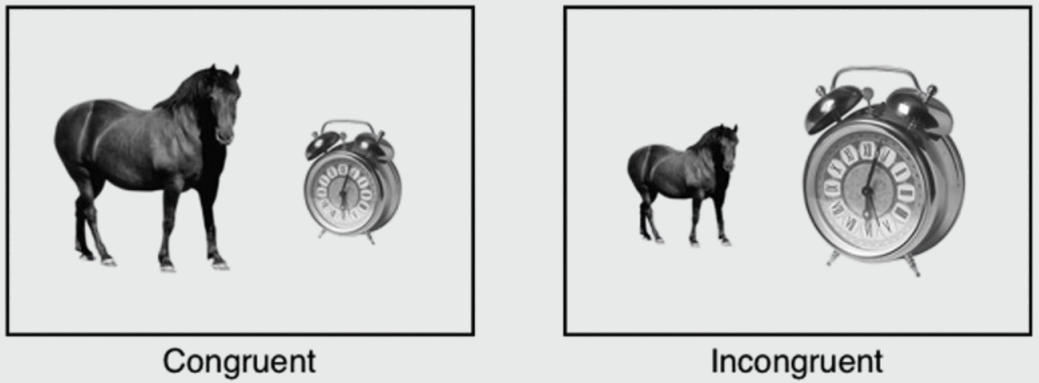
`tidyverse` provides several handy tools for bringing your data into the right format: **one variable per column, one observation per row**

Ideal situation: Think in advance how you want your dataset to look like, then program your experiment accordingly

Size Stroop task

Reminder: Participants are shown two images and they have to select the one that's smaller on the screen

Number of trials: 60 object pairs
× 2 trial types (congruent, incongruent)
× 2 presentation sides (horse-left, clock-right, horse-right, clock-left)



The data from one participant

browser	version	screenWidth	screenHeight	OS	OS_lang	GMT_timestamp	local_timestamp	mindsCode	link	calibration
Microsoft Edge	98.0.1108.43	1280	720	Windows	en-US	2022-02-09_09:44:06	2022-02-09_09:44:06	M057814	https://www.testable.org/experiment/4597/148338/start?participant=M057814	0.72
rowNo	type	stim1	stim2	stimPos	ITI	stimFormat	feedbackIncorrect	feedbackTime	head	responseType
2	form								What is the highest level of education you completed?	dropdown
190	test	croissantLarge	breadSmall	-341 0; 341 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
125	test	lampSmall	shelfLarge	-235 0; 235 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
110	test	hammerLarge	ladderSmall	-156 0; 156 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
215	test	wheelSmall	coinLarge	-300 0; 300 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
139	test	coconutSmall	blueberryLarge	-309 0; 309 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
160	test	pumpkinLarge	mushroomSmall	-304 0; 304 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
189	test	croissantSmall	breadLarge	-341 0; 341 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
114	test	trayLarge	bedSmall	-350 0; 350 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		
172	test	lettuceLarge	oliveSmall	-365 0; 365 0	700	.jpg	Oops, this is incorrect! \n Remember, choose the one which is smaller on the screen.	5000		

	age	sex	nationality																	
	41	male	british																	
	if	then	fixation	button1	keyboard	key	cursor	random	trialType	correctSide	task	stimPos_actual	ITI_ms	ITI_f	ITI_fDuration	timestamp	response	RT	correct	responseCode
r	Less than High School	screenout		Submit												24595	College or Technical School	3563	0	3
			ITI		f j	j	hide	2	incongruent	right	smaller	-341 0;341 0	700	42	16.67	63999	j	729	1	
			ITI		f j	f	hide	2	congruent	left	smaller	-235 0;235 0	700	42	16.67	65310	f	625	1	
			ITI		f j	j	hide	2	congruent	right	smaller	-370 0;370 0	700	42	16.67	67480	j	1477	1	
			ITI		f j	j	hide	2	congruent	right	smaller	-307 0;307 0	700	42	16.66	68963	j	793	1	
			ITI		f j	j	hide	2	incongruent	right	smaller	-156 0;156 0	700	42	16.66	70553	j	900	1	
			ITI		f j	f	hide	2	incongruent	left	smaller	-300 0;300 0	700	41	17.07	71952	f	699	1	
			ITI		f j	f	hide	2	incongruent	left	smaller	-309 0;309 0	700	42	16.66	73388	f	752	1	
			ITI		f j	j	hide	2	congruent	right	smaller	-304 0;304 0	700	42	16.67	74653	j	567	1	
			ITI		f j	f	hide	2	congruent	left	smaller	-341 0;341 0	700	42	16.67	75909	f	556	1	
			ITI		f j	j	hide	2	incongruent	right	smaller	-350 0;350 0	700	42	16.66	77404	j	801	1	

Importing the data: `read_csv`

The first thing you must do is import your dataset into R

```
data_path <- "data/148338_220209_095045_M057814.csv" # path to your csv file in a string
data <- read_csv(data_path, nskip=2) # nskip: How many rows to skip
```

This will create a *tibble*, which is the basic data structure in `tidyverse`: A table with named columns, each of which is a vector

rowNo	test	stim1	stim2	...	correct	responseCode
2	form	NA	NA		1	3
190	test	croissantLarge	breadSmall		1	NA
125	test	lampSmall	shelfLarge		1	NA
			...			

Selecting columns: `select`

Not all the columns in the current tibble are relevant for analyzing the data, so we might just as well remove them

`select`: keep or drop columns using their names

```
select(data, c(col1, col2, col3)) # Pass the column names to keep, no quotes needed

data %>% # Function chaining using the pipe operator
    select(c(col1, col2, col3))
    OR select(c(col1:col3)) # ':' keep all the columns in the range
    OR select(-c(col4:col5)) # remove columns by using '-'
```

`select` in action

In our example, we will want to keep all the information that will allow us to recreate what was presented on-screen on every trial as well as how the participant responded

```
data %>%
    select(c(rowNo, type, stim1, stim2, stimPos, trialType, response, RT))
```

A tibble: 241 x 8

rowNo	type	stim1	stim2	stimPos	trialType	response	RT
<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
2	form	NA	NA	NA	NA	College or Technical School	3563
190	test	croissantLarge	breadSmall	-341 0; 341 0	incongruent	j	729
125	test	lampSmall	shelfLarge	-235 0; 235 0	congruent	f	625
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	congruent	j	1477
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	congruent	j	793
110	test	hammerLarge	ladderSmall	-156 0; 156 0	incongruent	j	900

Selecting rows: `filter`

We want to also remove unnecessary rows (e.g., because data is missing, because the participant was not paying attention, etc.)

`filter`: keep rows that match a condition

```
filter(data, cond1, cond2, ...) # Keep only the rows that match your conditions

data %>%
    filter(trialType == 'incongruent') # Keep only the incongruent-trial rows
    OR filter(rt >= 200 & rt <= 1500) # Remove slow-response and fast-response trials
    OR filter(str_detect(col1, "croissant")) # Keep only the columns where "croissant" col1
    contains the substring 'croissant'
```

`filter` in action

In our example, we want to remove the first row, where information about the participant's education level is stored

```
data %>%
    select(c(rowNo, type, stim1, stim2, stimPos, trialType, response, RT)) %>%
    filter(type != "form")
```

A tibble: 240 x 8

rowNo	type	stim1	stim2	stimPos	trialType	response	RT
<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
190	test	croissantLarge	breadSmall	-341 0; 341 0	incongruent	j	729
125	test	lampSmall	shelfLarge	-235 0; 235 0	congruent	f	625
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	congruent	j	1477
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	congruent	j	793
110	test	hammerLarge	ladderSmall	-156 0; 156 0	incongruent	j	900
215	test	wheelSmall	coinLarge	-300 0; 300 0	incongruent	f	699

Columns are vectors

We now lost the info on participant's education level

Remember, that info is in the response column of the tibble

A tibble: 241 x 8

rowNo	type	stim1	stim2	stimPos	trialType	response	RT
<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
2	form	NA	NA	NA	NA	College or Technical School	3563
190	test	croissantLarge	breadSmall	-341 0; 341 0	incongruent	j	729
125	test	lampSmall	shelfLarge	-235 0; 235 0	congruent	f	625
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	congruent	j	1477
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	congruent	j	793
110	test	hammerLarge	ladderSmall	-156 0; 156 0	incongruent	j	900

We can extract it by *pulling* and *indexing* the vector response column

```
The base R way: response <- data$response; education_level <- response[1]
The tidyverse way: education_level <- data %>% pull(response) %>% first()
```

Adding columns: `mutate`

We also need to add columns for variables not yet present in the data

`mutate`: create, modify, or delete columns

```
mutate(data, new_col_1 = ..., new_col_2 = ...)

data %>%
    mutate(col6 = col4 * col5) # Create a new column storing the product of col4 and col5
    OR mutate(col7 = 'abc') # Constant value: All rows will have value 'abc'
    OR mutate(col8 = ifelse(stim1 == 'croissant', "French", NA)) # Conditional mutate
```

`mutate` in action: new columns

Let's add the education back and, using `row_number()`, a column containing information about the trial number

```
data %>%
    mutate(education = education_level) %>% # or directly: education = first(response)
    mutate(trial_number = row_number())
```

A tibble: 240 x 10

rowNo	type	stim1	stim2	stimPos	trialType	response	RT	education_level	trial_number
<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<chr>	<int>
190	test	croissantLarge	breadSmall	-341 0; 341 0	incongruent	j	729	College or Technical School	1
125	test	lampSmall	shelfLarge	-235 0; 235 0	congruent	f	625	College or Technical School	2
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	congruent	j	1477	College or Technical School	3
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	congruent	j	793	College or Technical School	4
110	test	hammerLarge	ladderSmall	-156 0; 156 0	incongruent	j	900	College or Technical School	5
215	test	wheelSmall	coinLarge	-300 0; 300 0	incongruent	f	699	College or Technical School	6
139	test	coconutSmall	blueberryLarge	-309 0; 309 0	incongruent	f	752	College or Technical School	7

`mutate` in action: modifying columns

When using a column name that already exists in the data, you will modify that column instead of creating a new one

```
data %>%
    mutate(trialType = factor(trialType, levels = c("incongruent", "congruent"))
```

A tibble: 240 x 10

rowNo	type	stim1	stim2	stimPos	trialType	response	RT	education_level	trial_number
<dbl>	<chr>	<chr>	<chr>	<chr>	<fct>	<chr>	<dbl>	<chr>	<int>
190	test	croissantLarge	breadSmall	-341 0; 341 0	incongruent	j	729	College or Technical School	1
125	test	lampSmall	shelfLarge	-235 0; 235 0	congruent	f	625	College or Technical School	2
200	test	schoolbusLarge	bikeSmall	-370 0; 370 0	congruent	j	1477	College or Technical School	3
152	test	orangeLarge	raspberrySmall	-307 0; 307 0	congruent	j	793	College or Technical School	4
110	test	hammerLarge	ladderSmall	-156 0; 156 0	incongruent	j	900	College or Technical School	5
215	test	wheelSmall	coinLarge	-300 0; 300 0	incongruent	f	699	College or Technical School	6
139	test	coconutSmall	blueberryLarge	-309 0; 309 0	incongruent	f	752	College or Technical School	7
160	test	pumpkinLarge	mushroomSmall	-304 0; 304 0	congruent	j	567	College or Technical School	8

Renaming columns: `rename`

We also need to add columns for variables not yet present in the data

`rename`: rename columns

```
rename(data, new_name = old_name)

data %>%
    rename(trial_type = trialType) # trialType → trial_type
```

Data tidying at a glance

```
education_level <- data %>% pull(response) %>% first()

data %>%
    # Keep only useful columns
    select(c(rowNo, type, stim1, stim2, stimPos, trialType, response, RT)) %>%
    # Keep only useful rows
    filter(type != "form") %>%
    # Add demographic and trial-number info, turn trial type to factor
    mutate(
        education_level = education_level,
        trial_number = row_number(),
        trialType = factor(trialType, levels = c("congruent", "incongruent"))
    ) %>%
    # Rename trialType to trial_type
    rename(trial_type = trialType)
```

Exercises

Open `Intro-Tidyverse.ipynb` and solve Exercises 1–11

The end result should look like this:

A tibble: 240 × 12

id	education_level	trial_block	trial_number	trial_type	stim_left	stim_right	stim_pos	correct_side	correct_key	rt	correct
<dbl>	<chr>	<dbl>	<dbl>	<fct>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>
1	College or Technical School	1	1	incongruent	croissantLarge	breadSmall	-341 0; 341 0	right	j	729	1
1	College or Technical School	1	2	congruent	lampSmall	shelfLarge	-235 0; 235 0	left	f	625	1
1	College or Technical School	1	3	congruent	schoolbusLarge	bikeSmall	-370 0; 370 0	right	j	1477	1
1	College or Technical School	1	4	congruent	orangeLarge	raspberrySmall	-307 0; 307 0	right	j	793	1
1	College or Technical School	1	5	incongruent	hammerLarge	ladderSmall	-156 0; 156 0	right	j	900	1
1	College or Technical School	1	6	incongruent	wheelSmall	coinLarge	-300 0; 300 0	left	f	699	1
1	College or Technical School	1	7	incongruent	coconutSmall	blueberryLarge	-309 0; 309 0	left	f	752	1
1	College or Technical School	1	8	congruent	pumpkinLarge	mushroomSmall	-304 0; 304 0	right	j	567	1
1	College or Technical School	1	9	congruent	croissantSmall	breadLarge	-341 0; 341 0	left	f	556	1
1	College or Technical School	1	10	incongruent	trayLarge	bedSmall	-350 0; 350 0	right	j	801	1
1	College or Technical School	1	11	congruent	lettuceLarge	oliveSmall	-365 0; 365 0	right	j	564	1
1	College or Technical School	1	12	incongruent	cornSmall	green beanLarge	-349 0; 349 0	left	f	597	1
1	College or Technical School	1	13	congruent	chilliSmall	eggplantLarge	-274 0; 274 0	left	f	534	1
1	College or Technical School	1	14	congruent	kittySmall	donkeyLarge	-309 0; 309 0	left	f	436	1

Summarizing the data: `summarize`

Now that the data is tidy, we can start analyzing it

`summarize`: summarize dataset to one row

```
summarize(data, f(col1)) # f is a function that takes vectors as input and outputs 1 value

data %>%
    summarize(mean(col1), sd(col2), sum(col3)) # average of col1, st-dev of col2, etc.
    OR summarize(mu = mean(col1), std = sd(col2), sigma = sum(col3)) # new columns can be
    named
```

Summarizing a grouped data: `group_by`

What if we want to get descriptive statistics for each level of a variable?

`summarize`: summarize each **group** to one row

```
summarize(data, f(col1), .by = col5) # will return as many rows as col5 has unique values

data %>%
    summarize(mean(col1), .by = col5) # average of col1 at each level of col5

# Equivalently
data %>%
    group_by(col5) %>%
    summarize(mean(col1)) %>%
    ungroup()
```

`summarize` in action

```
tidy_data %>%
    summarize(rt = mean(rt), accuracy = mean(correct), error = mean(1 - correct))
```

A tibble: 1 × 3

rt	accuracy	error
<dbl>	<dbl>	<dbl>
622.7542	0.9833333	0.01666667

```
tidy_data %>%
    summarize(rt = mean(rt), .by = trial_type)
```

A tibble: 2 × 2

trial_type	rt
<fct>	<dbl>
incongruent	698.1000
congruent	547.4083

`mutate` w/ summary functions

Summary functions (with or without grouping) can also be used to create new columns

```
tidy_data %>%
    mutate(avg_error = mean(1 - correct))
```

id	education_level	trial_block	trial_number	trial_type	stim_left	stim_right	stim_pos	correct_side	correct_key	rt	correct	avg_error
<dbl>	<chr>	<dbl>	<dbl>	<fct>	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
1	College or Technical School	1	1	incongruent	croissantLarge	breadSmall	-341 0; 341 0	right	j	729	1	0.01666667
1	College or Technical School	1	2	congruent	lampSmall	shelfLarge	-235 0; 235 0	left	f	625	1	0.01666667
1	College or Technical School	1	3	congruent	schoolbusLarge	bikeSmall	-370 0; 370 0	right	j	1477	1	0.01666667
1	College or Technical School	1	4	congruent	orangeLarge	raspberrySmall	-307 0; 307 0	right	j	793	1	0.01666667
1	College or Technical School	1	5	incongruent	hammerLarge	ladderSmall	-156 0; 156 0	right	j	900	1	0.01666667

Checking counterbalancing w/ `tidyverse`

Grouping together with the `n()` function allows you to easily spot whether the experiment was counterbalanced

```
tidy_data %>%
     summarize(n = n(), .by = c(trial_type, correct_key))
```

A tibble: 4 × 3

trial_type	correct_key	n
<fct>	<chr>	<int>
incongruent	j	60
congruent	f	60
congruent	j	60
incongruent	f	60

Summary

To get your data in good shape, you need to:

1) **Understand the goal**: 1 variable per column, 1 observation per row
2) **Master a handful of functions**: `select`, `filter`, `mutate`, `summarize`
3) **Understand grouping**: `.by` (argument to mutate, filter, summarize) or `group_by` (standalone function)

Interlude: An example of a mess

How I organized the data for my first experiment…

AGE	CONDITION	TRIAL ORDER	BOX SETUP	EXP POSITION	1.CHOICE	1.CORRECT	1.SIDE	2.CHOICE	2.CORRECT	2.SIDE	3.CHOICE	3.CORRECT	3.SIDE	4.CHOICE	4.CORRECT	4.SIDE
20;7	REAL	LR-LRRL	ORANGE - BLUE	LLRR	1	1	0	1	0	0	0	NA	NA	1	1	0
19;14	REAL	LR-LRRL	ORANGE - BLUE	LLRR	1	1	0	1	1	1	1	1	1	1	1	0
19;13	REAL	LR-LRRL	ORANGE - BLUE	RRLL	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA
19;28	REAL	LR-LRRL	ORANGE - BLUE	RRLL	1	1	0	1	1	1	0	NA	NA	1	1	0
19;21	REAL	LR-LRRL	BLUE - ORANGE	LLRR	1	1	0	1	1	1	1	1	1	1	1	0
19;3	REAL	LR-LRRL	BLUE - ORANGE	LLRR	1	1	0	1	1	1	1	1	1	1	1	0
19;28	REAL	LR-LRRL	BLUE - ORANGE	RRLL	1	1	0	1	0	0	1	0	0	1	1	0
19;27	REAL	LR-LRRL	BLUE - ORANGE	RRLL	1	1	0	1	1	1	1	1	1	1	1	0
19;7	REAL	RL-RLLR	ORANGE - BLUE	LLRR	1	1	1	1	1	0	1	1	0	1	0	0
19;8	REAL	RL-RLLR	ORANGE - BLUE	LLRR	1	1	1	1	1	0	1	1	0	1	1	1
19;26	REAL	RL-RLLR	ORANGE - BLUE	RRLL	0	NA	NA	0	NA	NA	NA	NA	NA	1	1	1
19;6	REAL	RL-RLLR	ORANGE - BLUE	RRLL	0	NA	NA	0	NA	NA	1	0	1	1	0	0
18;20	REAL	RL-RLLR	BLUE - ORANGE	LLRR	1	1	1	1	1	0	0	NA	NA	0	NA	NA
19;9	REAL	RL-RLLR	BLUE - ORANGE	LLRR	1	1	1	NA	NA	NA	0	NA	NA	0	NA	NA
19;4	REAL	RL-RLLR	BLUE - ORANGE	RRLL	0	NA	NA	1	1	0	1	1	0	NA	NA	NA
19;12	REAL	RL-RLLR	BLUE - ORANGE	RRLL	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA

An example of a mess

…instead of:

id	trial_block	trial_number	correct_answer	box_left	experimenter_pos	choice	correct
1	familiarization	1	left	orange	center	left	1
1	familiarization	2	right	orange	center	right	1
1	test	1	left	orange	left	left	1
1	test	2	right	orange	left	right	1
1	test	3	right	orange	right	NA	NA
1	test	4	left	orange	right	left	1
			…				

From one to multiple participants

Exercises 15–18 in `Intro-Tidyverse.ipynb` recapitulate the steps we have taken so far but for a full dataset (12 subjects)

Caveat: You need to be careful when adding variables

> When adding the trial-number column, for instance, calling `row_number()` without grouping will provide numbers for the entire table (1–2880) instead of 1–240 for id 1, 1–240 for id 2…
>
> Same for creating the `education_level` variable: When calling `first(response)`, the data needs to be grouped by subject

Changing the format of the data

An example: Suppose we want to obtain the numerical Stroop effect (Incongruent RTs – Congruent RTs) for each subject

This won't do because we can only perform operations over columns, while trial type information is in rows

```
full_data %>%
    group_by(id, trial_type) %>%
    summarize(rt = mean(rt)) %>%
    ungroup()
```

id	trial_type	rt
<chr>	<fct>	<dbl>
1	congruent	547.4083
1	incongruent	683.9576
10	congruent	792.3714
10	incongruent	982.6264
11	congruent	587.2857
11	incongruent	624.1949
12	congruent	736.6887
12	incongruent	944.8113
2	congruent	722.7315
2	incongruent	738.2407
3	congruent	677.9316
3	incongruent	804.0000
5	congruent	706.3913
5	incongruent	862.4248
6	congruent	720.4474
6	incongruent	835.8704
7	congruent	909.7333
7	incongruent	1003.2118
8	congruent	532.4790
8	incongruent	574.1186
9	congruent	1003.9762
9	incongruent	1109.5844

Spreading variables over columns: `pivot_wider`

Solution: Move from trial type as one column with two values to two columns with reaction-time values by using `pivot_wider`

id	trial_type	rt
<chr>	<fct>	<dbl>
1	congruent	547.4083
1	incongruent	683.9576
10	congruent	792.3714
10	incongruent	982.6264
11	congruent	587.2857
11	incongruent	624.1949
12	congruent	736.6887
12	incongruent	944.8113
2	congruent	722.7315
2	incongruent	738.2407
3	congruent	677.9316
3	incongruent	804.0000
5	congruent	706.3913
5	incongruent	862.4248
6	congruent	720.4474
6	incongruent	835.8704
7	congruent	909.7333
7	incongruent	1003.2118
8	congruent	532.4790
8	incongruent	574.1186
9	congruent	1003.9762
9	incongruent	1109.5844

→

```
long_data %>%
   pivot_wider(
     names_from = trial_type,
     values_from = rt
     )
```

→

id	incongruent	congruent
<chr>	<dbl>	<dbl>
1	683.9576	547.4083
2	738.2407	722.7315
3	804.0000	677.9316
5	862.4248	706.3913
6	835.8704	720.4474
7	1003.2118	909.7333
8	574.1186	532.4790
9	1109.5844	1003.9762
10	982.6264	792.3714
11	624.1949	587.2857
12	944.8113	736.6887

→

```
wide_data %>%
   mutate(stroop=inc - con) %>%
   arrange(stroop)
```

→

id	stroop
<chr>	<dbl>
2	15.50926
11	36.90920
8	41.63965
7	93.47843
9	105.60823
6	115.42300
3	126.06838
1	136.54929
5	156.03347
10	190.25495
12	208.12264

Gathering variables in rows: `pivot_longer`

The opposite of `pivot_wider`: `pivot_longer`

id	stroop_rt	stroop_error
<chr>	<dbl>	<dbl>
1	136.54929	0.03389831
10	190.25495	0.06739927
11	36.90920	0.10176613
12	208.12264	0.03773585
2	15.50926	0.06481481
3	126.06838	0.08547009
5	156.03347	0.10619469
6	115.42300	0.02923977
7	93.47843	0.02352941
8	41.63965	0.05939325
9	105.60823	0.01298701

```
long_data %>%
    pivot_longer(
      cols = c(stroop_rt, stroop_error),
      names_to = "measure",
      values_to = "stroop_value"
      )
```

id	measure	value
<chr>	<chr>	<dbl>
1	stroop_rt	136.54929379
1	stroop_error	0.03389831
10	stroop_rt	190.25494505
10	stroop_error	0.06739927
11	stroop_rt	36.90920097
11	stroop_error	0.10176613
12	stroop_rt	208.12264151
12	stroop_error	0.03773585
2	stroop_rt	15.50925926
2	stroop_error	0.06481481
3	stroop_rt	126.06837607
3	stroop_error	0.08547009
5	stroop_rt	156.03347441
5	stroop_error	0.10619469
6	stroop_rt	115.42300195
6	stroop_error	0.02923977
7	stroop_rt	93.47843137
7	stroop_error	0.02352941
8	stroop_rt	41.63965247
8	stroop_error	0.05939325
9	stroop_rt	105.60822511
9	stroop_error	0.01298701

Final assignment

Solve Exercises 12–20 in `Intro-Tidyverse.ipynb`

Data plotting basics

Underlying philosophy

The grammar of graphics: A way of thinking about graphs as compositional objects constructed out of primitive building blocks

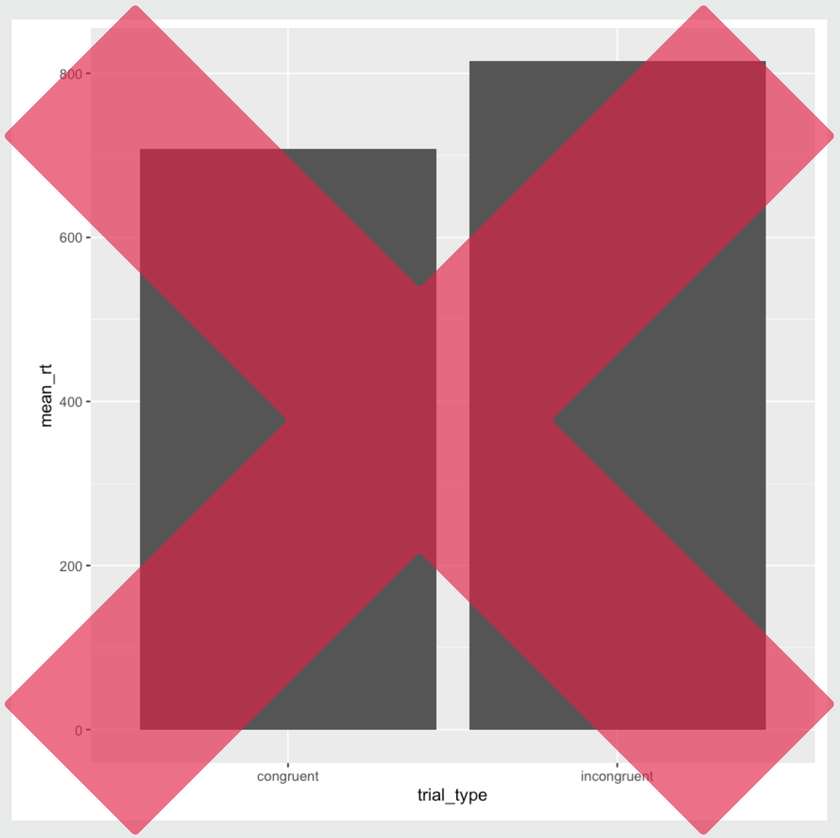
1. **Data** to plot
2. **Aesthetics**: Which variables are mapped onto which visual properties (e.g., reaction times onto *y*-axis, trial type onto color)
3. **Geoms**: Visual marks used to represent data (boxplots, histograms)
4. **Stats**: Statistical transformations (means, confidence intervals)

There are a few others, but these are the most important ones: To learn more, go here

Two guiding principles

Principle 1: If your data is tidy, plotting with `ggplot` is a piece of cake

Principle 2: No barplots!



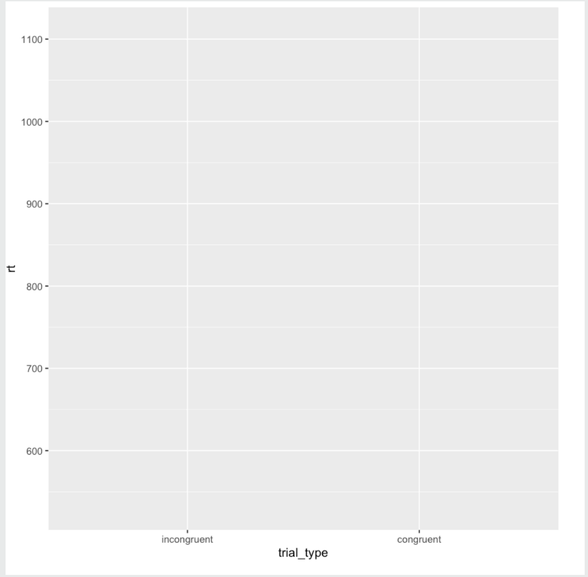
Building a plot step-wise

First, let's average the data by id and trial type

```
average_data <- full_data %>% summarize(rt = mean(rt), .by = c(id, trial_type))
```

Let's initialize the plot

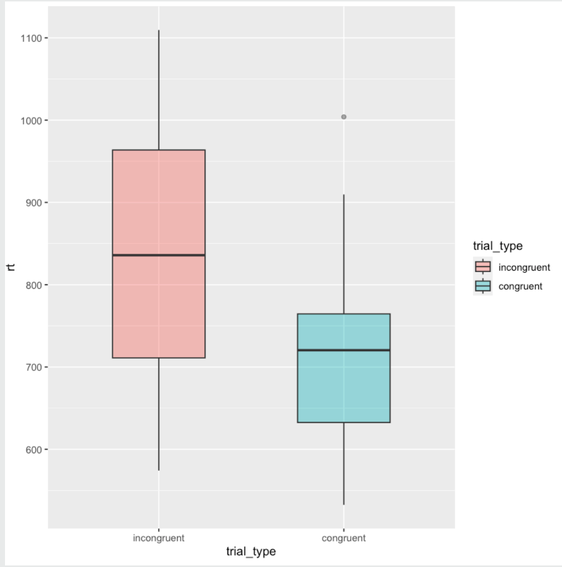
```
# Creates the plot object and defines the data and aesthetics
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type))
```



Building a plot step-wise

Now we can add the first `geom`: **a boxplot** (note the '+' sign)

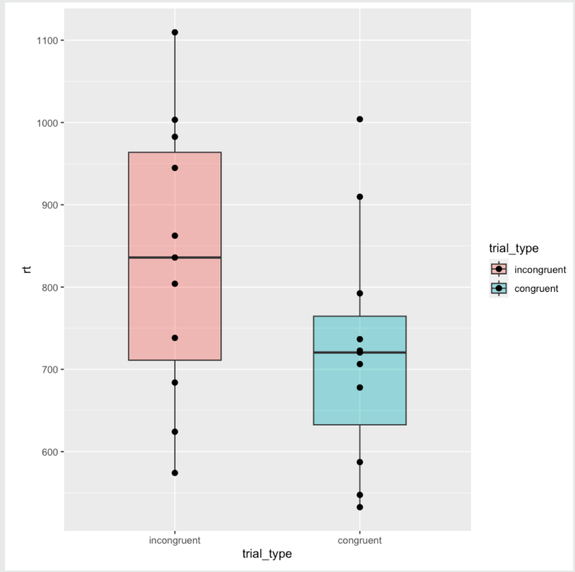
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
   # Create a box plot
   geom_boxplot(width=0.5, alpha=0.45)
```



Building a plot step-wise

We can now add a second `geom`: **individual datapoints**

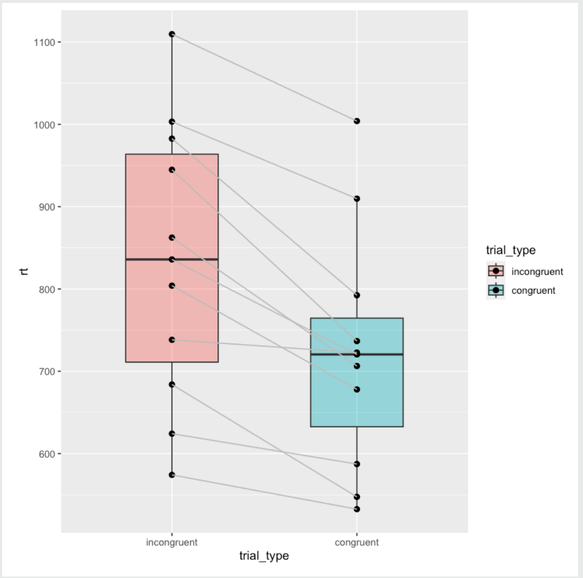
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
    # Create a box plot
    geom_boxplot(width=0.5, alpha=0.45) +
    # Add individual datapoints based on x- and y-axes
    geom_point(size=2)
```



Building a plot step-wise

A third `geom`: **individual variability**

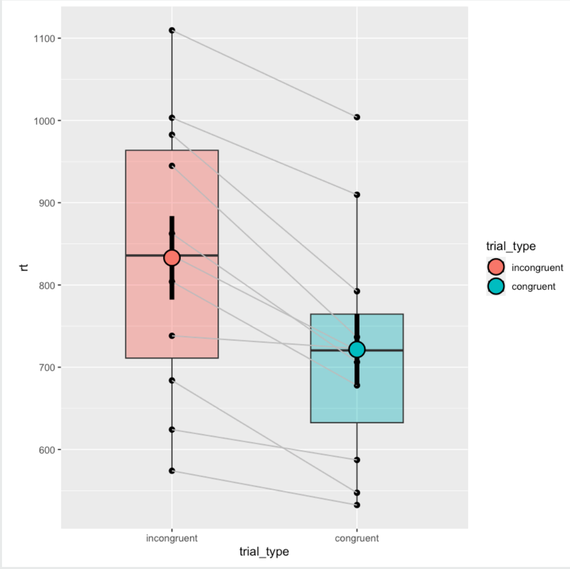
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
    # Create a box plot
    geom_boxplot(width=0.5, alpha=0.45) +
    # Add individual datapoints based on x- and y-axes
    geom_point(size=2) +
    # Connect them to show how effect consistency
    geom_line(aes(group=id), color='gray')
```



Building a plot step-wise

Adding a `stat`: group means and standard errors

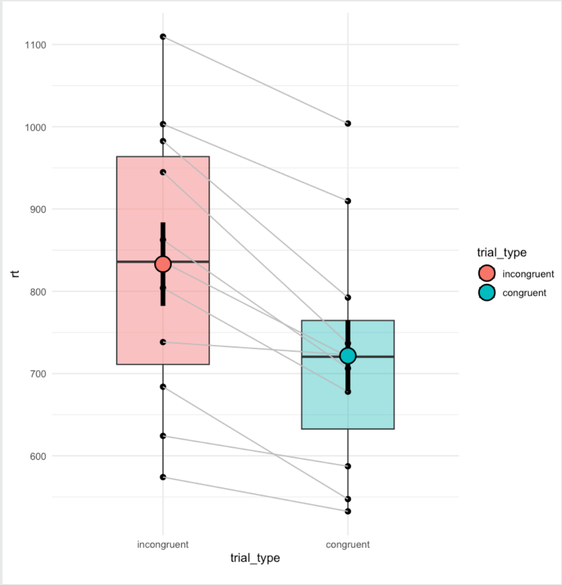
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
    # Create a box plot
    geom_boxplot(width=0.5, alpha=0.45) +
    # Add individual datapoints based on x- and y-axes
    geom_point(size=2) +
    # Connect them to show how effect consistency
    geom_line(aes(group=id), color='gray') +
    # Add mean and SEM info
    stat_summary(fun.data=mean_se, linewidth=2, shape=21, size=1.5)
```



Building a plot step-wise

Discard the default theme to another preset: `theme_minimal()`

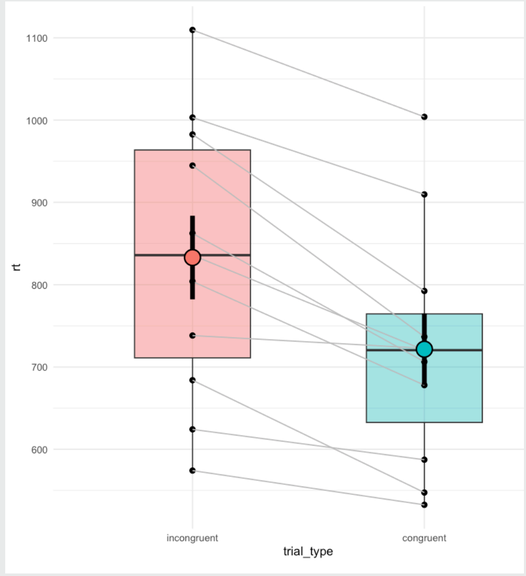
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
   # Create a box plot
   geom_boxplot(width=0.5, alpha=0.45) +
   # Add individual datapoints based on x- and y-axes
   geom_point(size=2) +
   # Connect them to show how effect consistency
   geom_line(aes(group=id), color='gray') +
   # Add mean and SEM info
   stat_summary(fun.data=mean_se, linewidth=2, shape=21, size=1.5) +
   theme_minimal() # Change from default theme to a better one
```



Building a plot step-wise

Removing the redundant legend

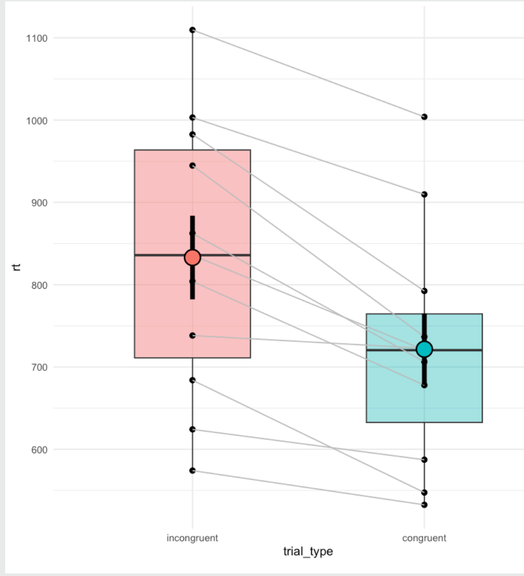
```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
   # Create a box plot
   geom_boxplot(width=0.5, alpha=0.45) +
   # Add individual datapoints based on x- and y-axes
   geom_point(size=2) +
   # Connect them to show how effect consistency
   geom_line(aes(group=id), color='gray') +
   # Add mean and SEM info
   stat_summary(fun.data=mean_se, linewidth=2, shape=21, size=1.5) +
   theme_minimal() + # Change from default theme to a better one
   theme(legend.position="none") # Remove legend
```



Building a plot step-wise

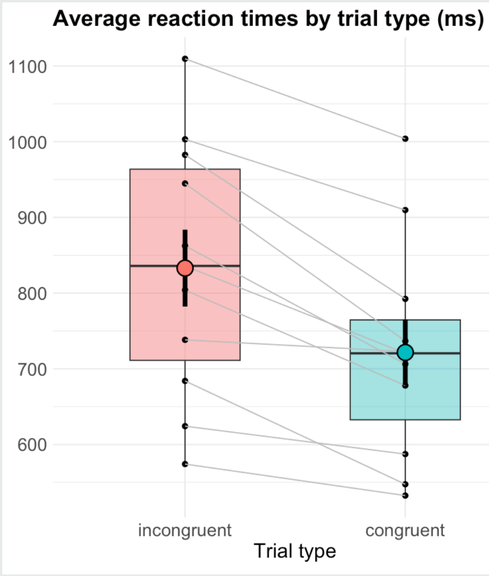
Improve the labels

```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
   # Create a box plot
   geom_boxplot(width=0.5, alpha=0.45) +
   # Add individual datapoints based on x- and y-axes
   geom_point(size=2) +
   # Connect them to show how effect consistency
   geom_line(aes(group=id), color='gray') +
   # Add mean and SEM info
   stat_summary(fun.data=mean_se, linewidth=2, shape=21, size=1.5) +
   theme_minimal() + # Change from default theme to a better one
   theme(legend.position="none") + # Remove legend
   labs(title = "Average reaction times by trial type (ms)", x = "Trial
   type", y = "") # Change labels
```



The full plot

```
ggplot(average_data, aes(x = trial_type, y = rt, fill = trial_type)) +
   geom_boxplot(width=0.5, alpha=0.45) + # Create a box plot
   # Add individual datapoints based on x- and y-axes
   geom_point(size=2) +
   # Connect them to show how effect consistency
   geom_line(aes(group=id), color='gray') +
   # Add mean and SEM info
   stat_summary(fun.data=mean_se, linewidth=2, shape=21, size=1.5) +
   # Customize appearance
   theme_minimal() + # Change from default theme to a better one
   theme(legend.position="none", plot.title=element_text(face = "bold",
   size = 20), axis.title=element_text(size = 18),
   axis.text=element_text(size = 16)) + # Remove legend, change text size
   # Change labels
   labs(title="Average RTs by trial type (ms)", x="Trial type", y = "")
```



Push your work to GitHub

Give us feedback

https://forms.gle/6brjKZs2esUuGqkG8